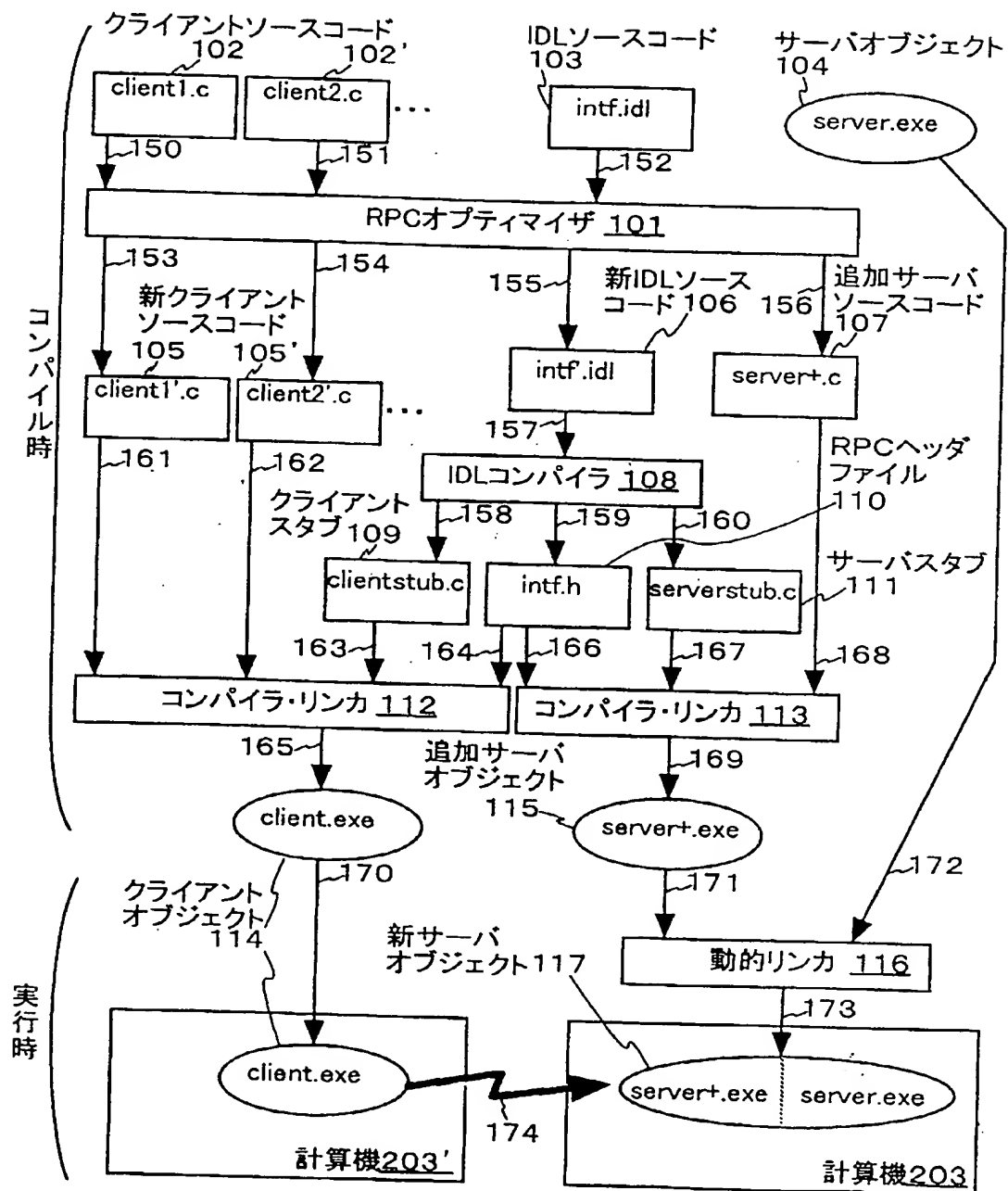


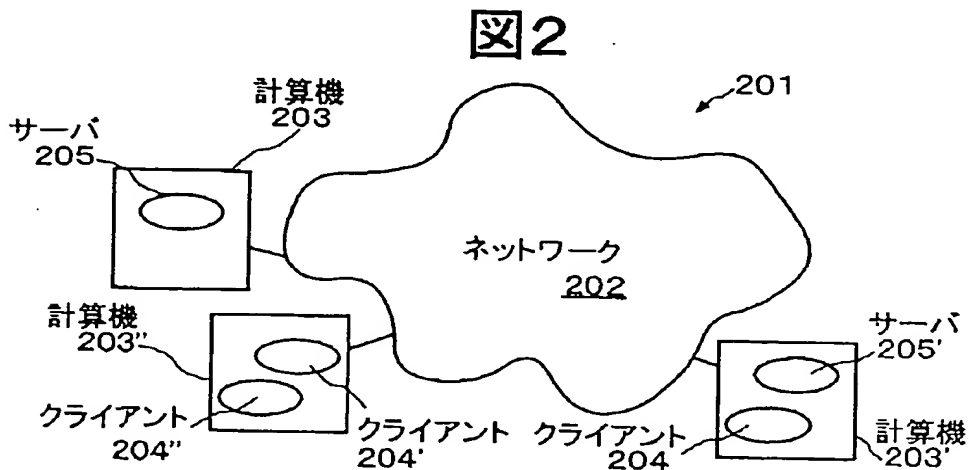
【書類名】 図面

【図1】

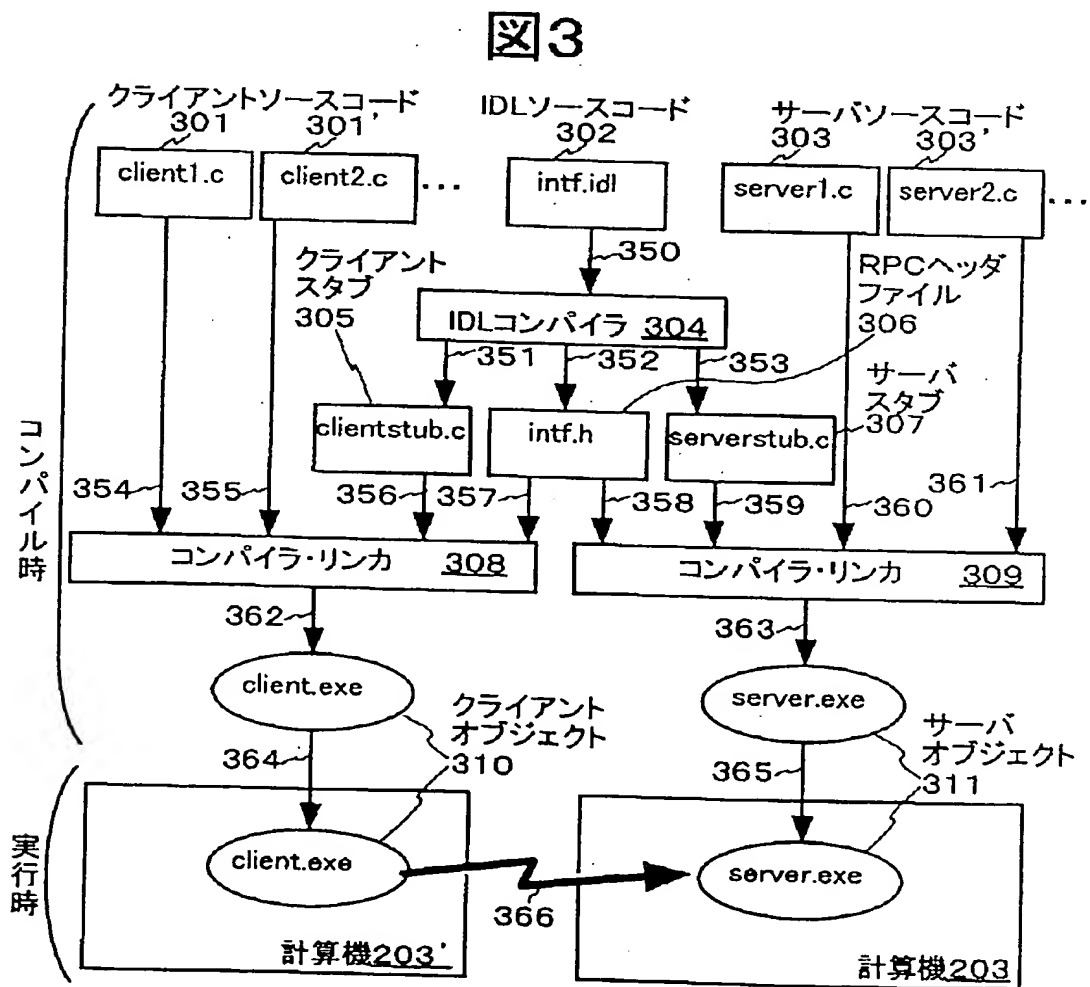
図1



【図 2】

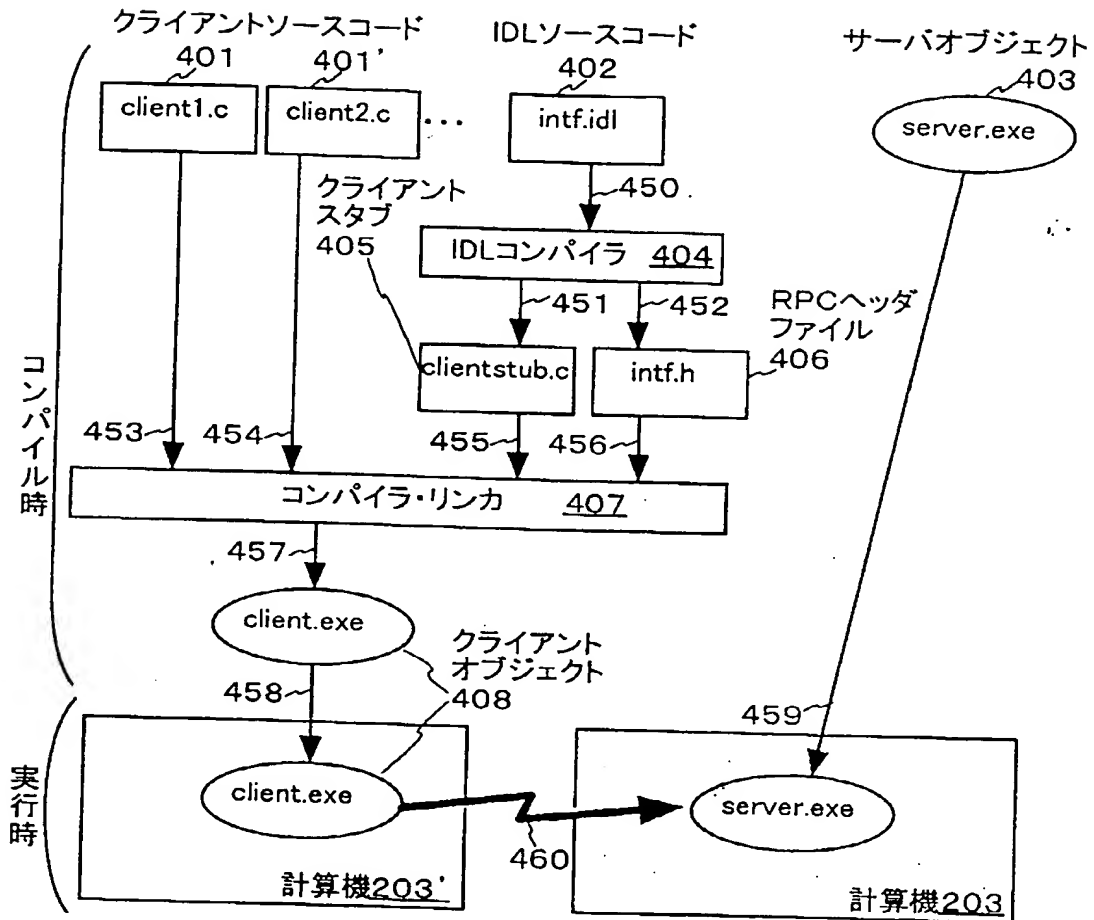


【図 3】



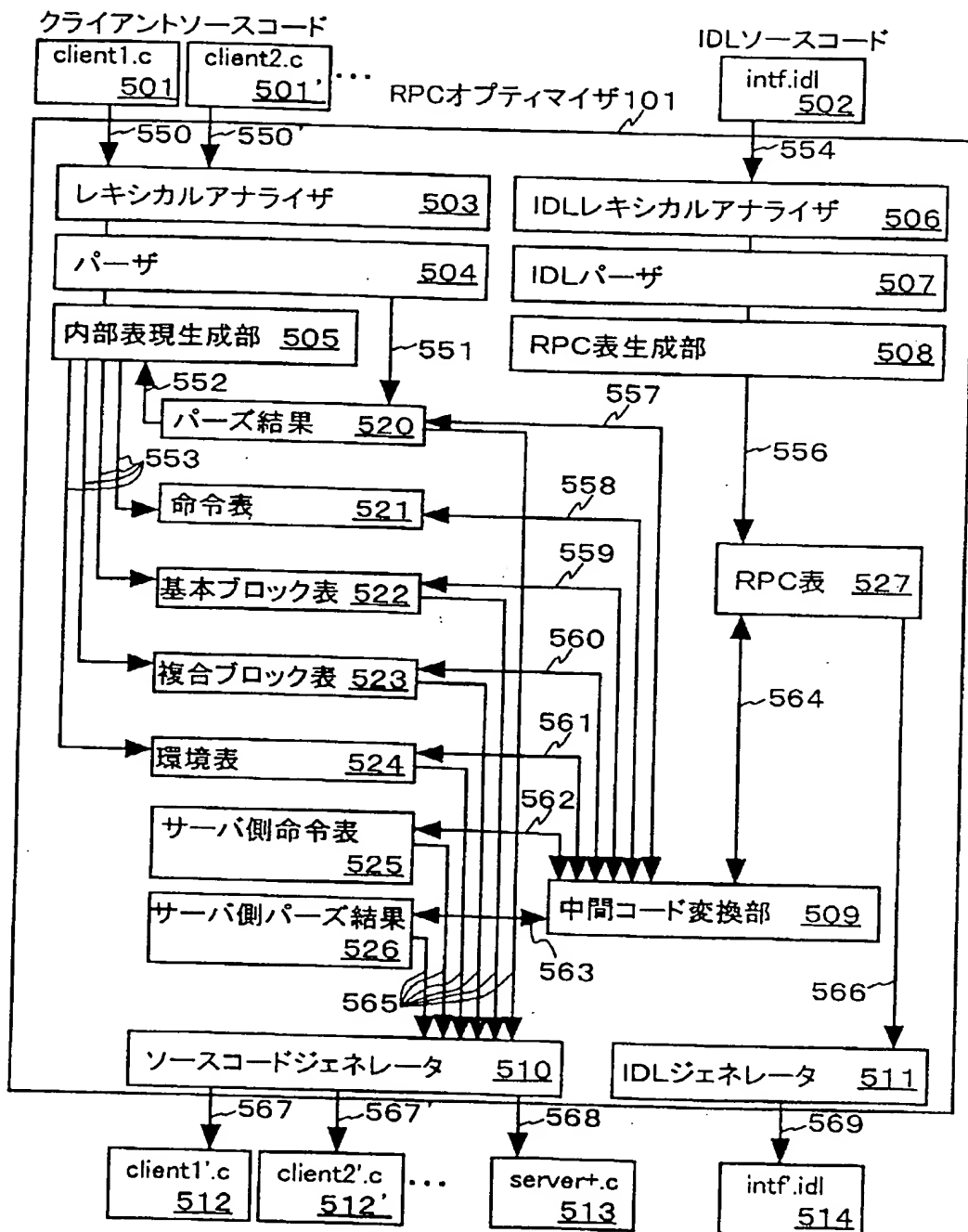
【図4】

図4



【図5】

図5



【図6】

# 図6

## 命令表 600

命令ID 602	ターゲット 603	命令 604	オペランドA 605	オペランドB 606
命令要素 601 ✓				
⋮				

## 基本ブロック表 610

基本ブロックID 612	開始命令ID 613	終端命令ID 614
次基本ブロック 615	前基本ブロック 616	環境ID 617
DGEN変数表 618	DKILL変数表 619	DIN変数表 620
DOUT変数表 621	LIN変数表 622	LOUT変数表 623
LUSE変数表 624	LDEF変数表 625	
基本ブロック要素 611 ✓		
⋮		

## 複合ブロック表 630

複合ブロックID 632	開始基本ブロックID 633	終端基本ブロックID 634	環境ID 635
複合ブロック要素 631 ✓			
⋮			

## 環境表 640

環境ID 641	親環境ID 642	属性 643
環境内変数表 644		

## RPC表 650

RPC名 652	IN引数表 653	OUT引数表 654	属性 655
RPC表要素651✓			
⋮			
型名 656	型情報 657		
型宣言要素658✓			
⋮			

## 変数表 660

変数名 662	型 663	属性 664
変数表要素 661 ✓		
⋮		

66260"6656460

[illegible]

图 7

700

700

750

750

【図 8】

図 8

intf.h

```
801 #include "Object.h"
802 class MyServer: public Object {
803     int func1(int i);
804     void func2(long& key, char* value);
805 }
```

800

clientstub.c

```
851 #include "intf.h"
852 int MyServer::func1(int i)
853 {
854     Buffer buf = new Buffer();
855     int rval;
856     buf.packint(i);
857     call("func1", buf);
858     buf.unpackint(&rval);
859     delete buf;
860     return rval;
861 }
862 void MyServer::func2(long& key, char* value)
863 {
864     Buffer buf = new Buffer();
865     buf.packlong(key);
866     buf.packString(value);
867     call("func2", buf);
868     buf.unpacklong(&key);
869     delete buf;
870 }
```

850

64250" 6350160

【図9】

図9

```
serverstub.c
901 #include "intf.h"
902 void MyServer::loop()
903 {
904     while (1) {
905         Buffer buf;
906         Client client;
907         receive(&client, &buf);
908         if (buf.method.equals("func1")) {
909             int i, rval;
910             buf.unpackint(&i);
911             rval = func1(i);
912             buf.packint(rval);
913         } else if (buf.method.equals("func2")) {
914             long key;
915             char* value;
916             buf.unpacklong(&key);
917             buf.unpackString(&value);
918             func2(key, value);
919             buf.packlong(key);
920         } else {
921             send(client, "error");
922             continue;
923         }
924         send(client, buf);
925         delete buf;
926         delete client;
927     }
928 }
```

900

66260"6858160



图 10

intf.idl

```
1001 interface MyServer {
1002     int func1(in int i);
1003     void func2(inout long key, in String value);
1004     void func3(inout int count);
1005     void func4(in int i);
1006 };
```

1000

client1.c

```
1011 #include "intf.h"
1012 main()
1013 {
1014     MyServer server = lookupDirectory("MyServer");
1015     int count = 0;
1016     server.func3(count);
1017     printf("count=%d\n", count);
1018     server.func4(j);
1019 }
```

1010

server.c

```

1031 #include "intf.h"

1032 void MyServer::func3(int& count)
1033 {
1034     for (int i = 0; i < 100; i++)
1035         count += server.func1(i);
1036 }

1037 void MyServer::func4(int count)
1038 {
1039     server.func2(100, "hello world");
1040     server.func1(count);
1041 }

```

1030

【図 11】

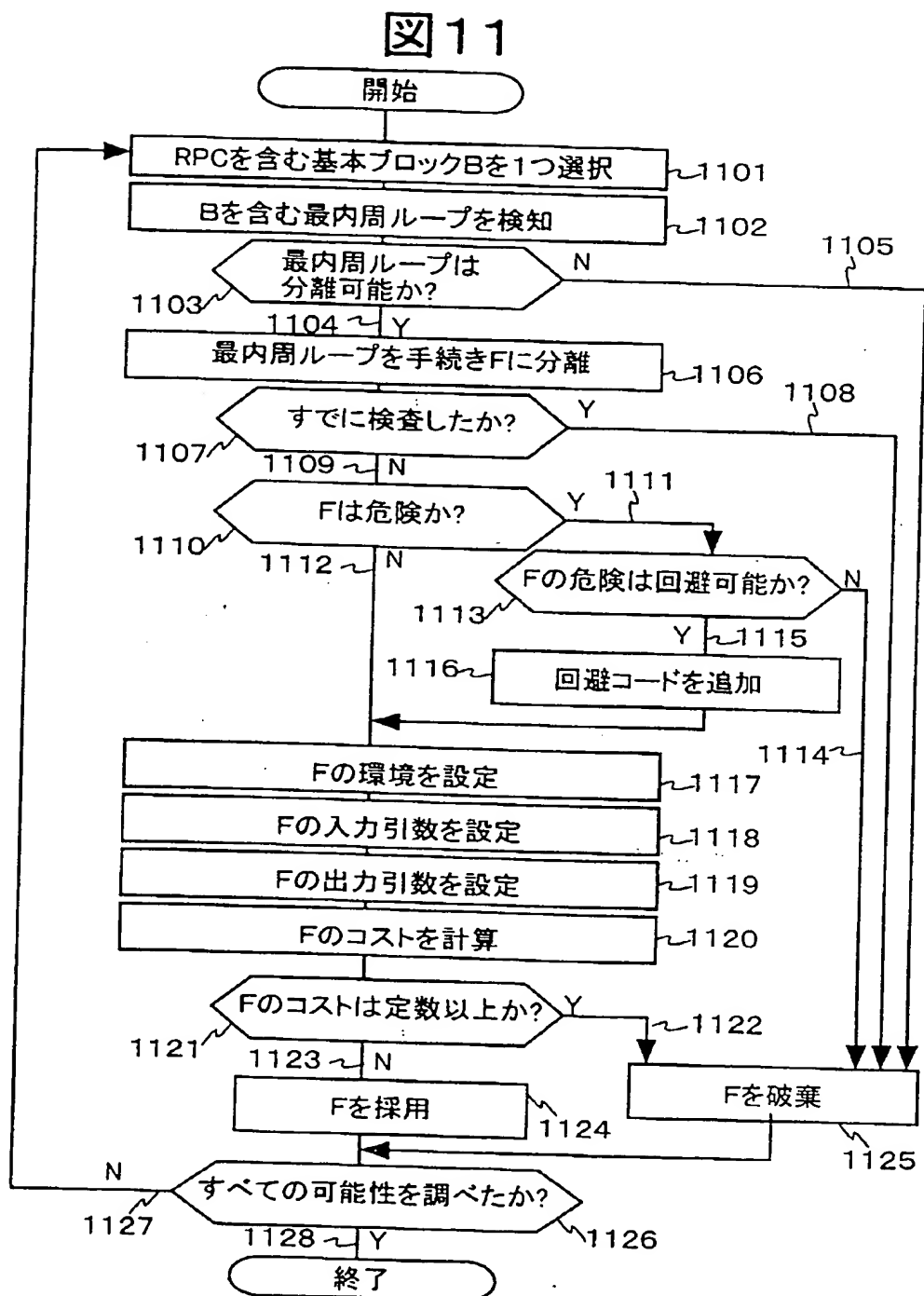
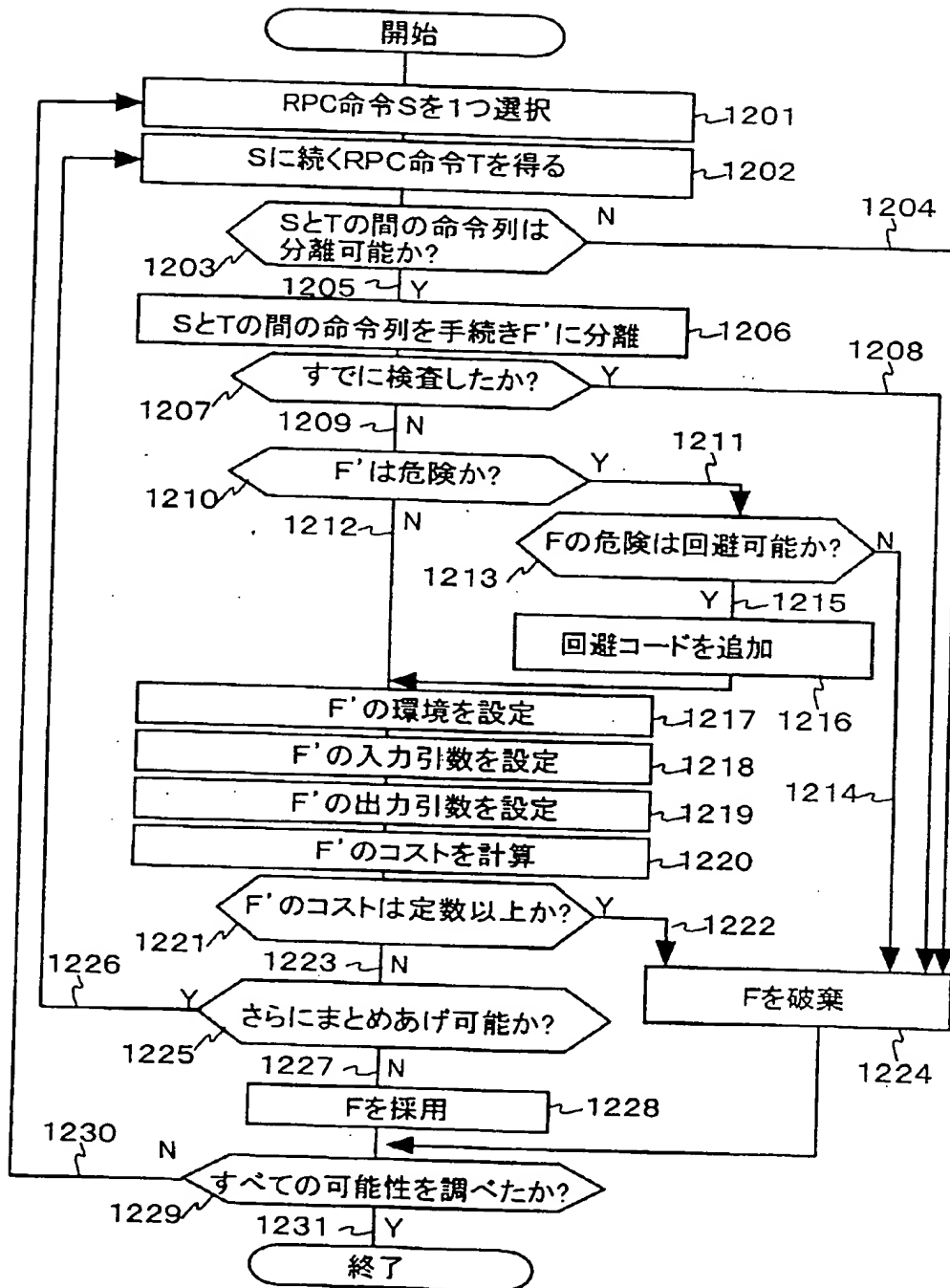
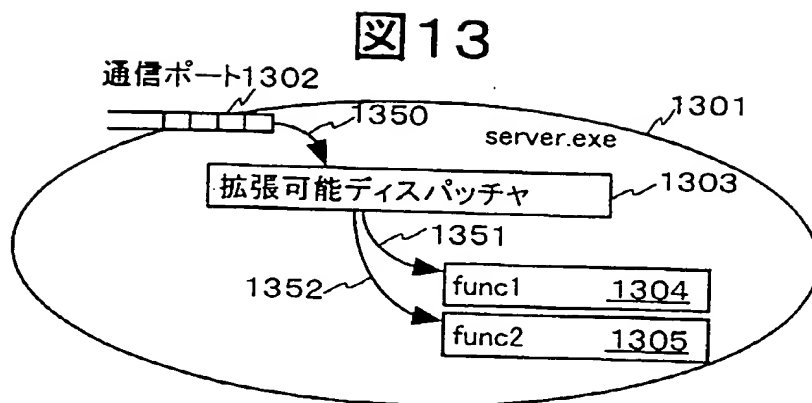


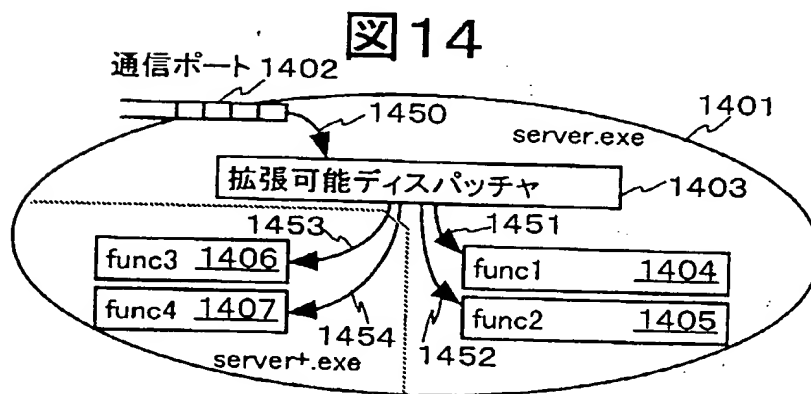
図12



【図13】

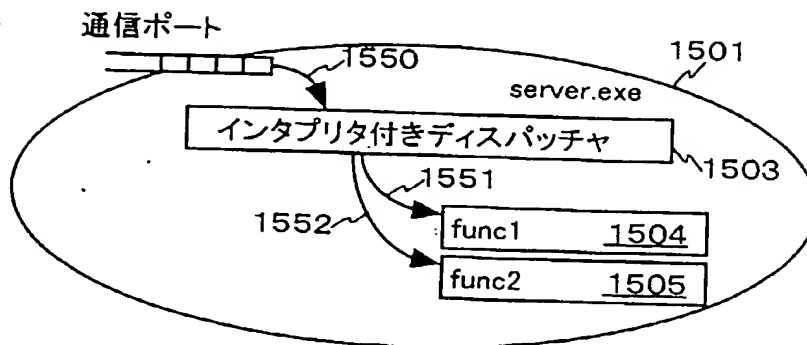


【図14】



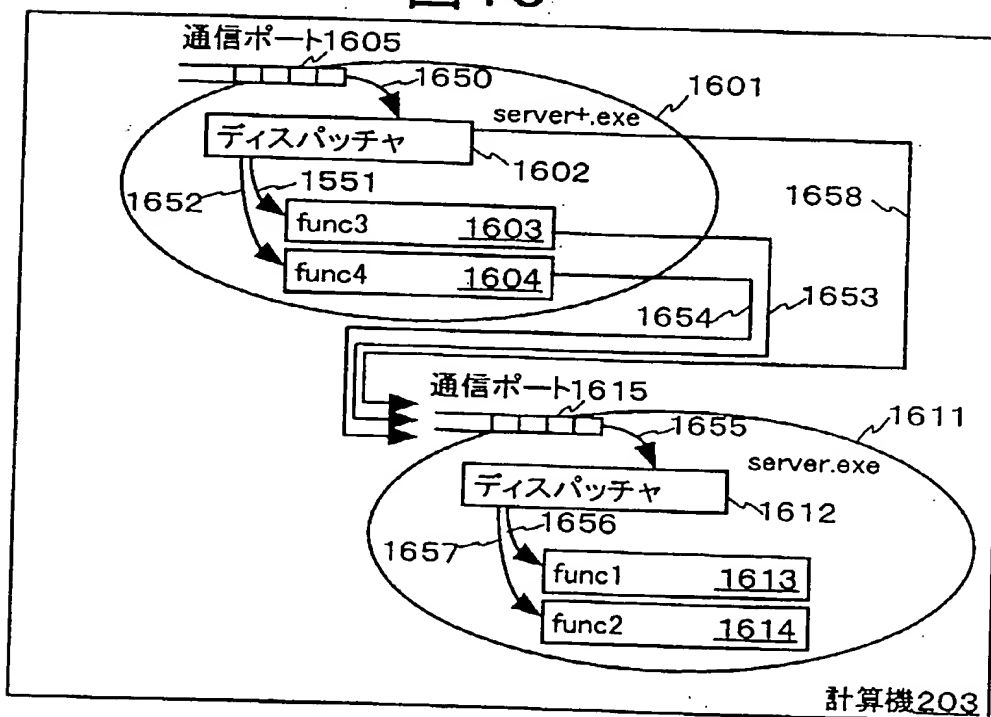
【図15】

図15



【図16】

図16



【図17】

図17

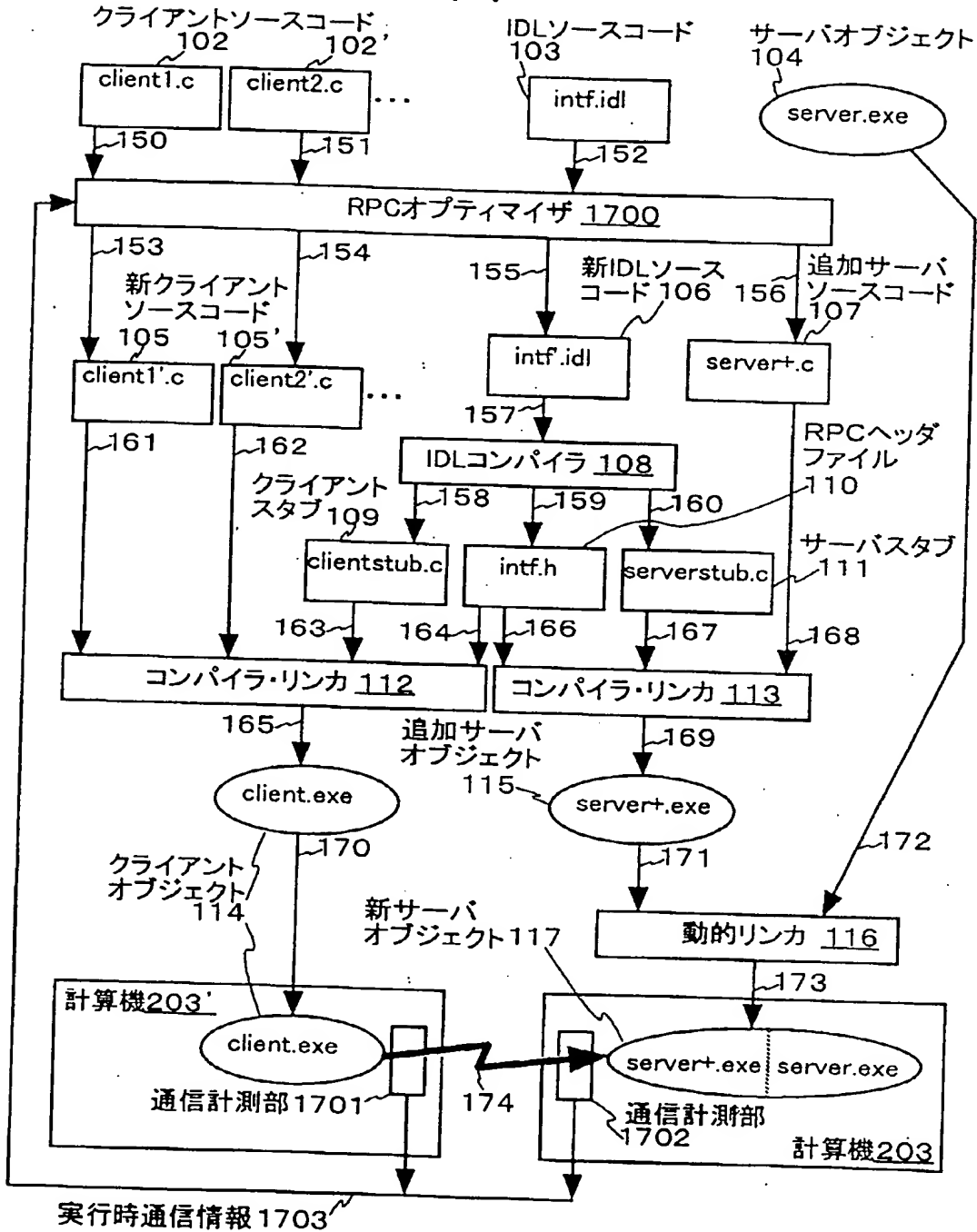


图 18

extended intf.idl

```
1801 interface MyServer {
1802     int func1(in int i) const;
1803     void func2(inout long key, in String value);
1804     int func3(void);
1805     commutative { func2, func3 };
1806     parallel { func1, func2, func3 };
1807 };
```

1800

server.c

```

1821 #include "intf.h"
1822 #include "thread.h"

1823 void MyServer::func3(int& count)
1824 {
1825     List<Thread> allThreads;
1826     Thread t;
1827     void *rval;
1828     for (int i = 0; i < 100; i++) {
1829         create_thread(&t, server.func1, 1, i);
1830         allThreads.add(t);
1831     }
1832     for ( ; (t = allThreads.next()) != NULL_THREAD; ) {
1833         join_thread(t, &rval);
1834         count += *(int *)rval;
1835     }
1836 }

1837 void MyServer::func4(int count)
1838 {
1839     List<Thread> allThreads;
1840     Thread t;
1841     create_thread(&t, server.func2, 2, 100, "hello world");
1842     allThreads.add(t);
1843     create_thread(&t, server.func1, 1, count);
1844     allThreads.add(t);
1845     for ( ; (t = allThreads.next()) != NULL_THREAD; )
1846         join_thread(t, NULL);
1847 }

```

1820

図19

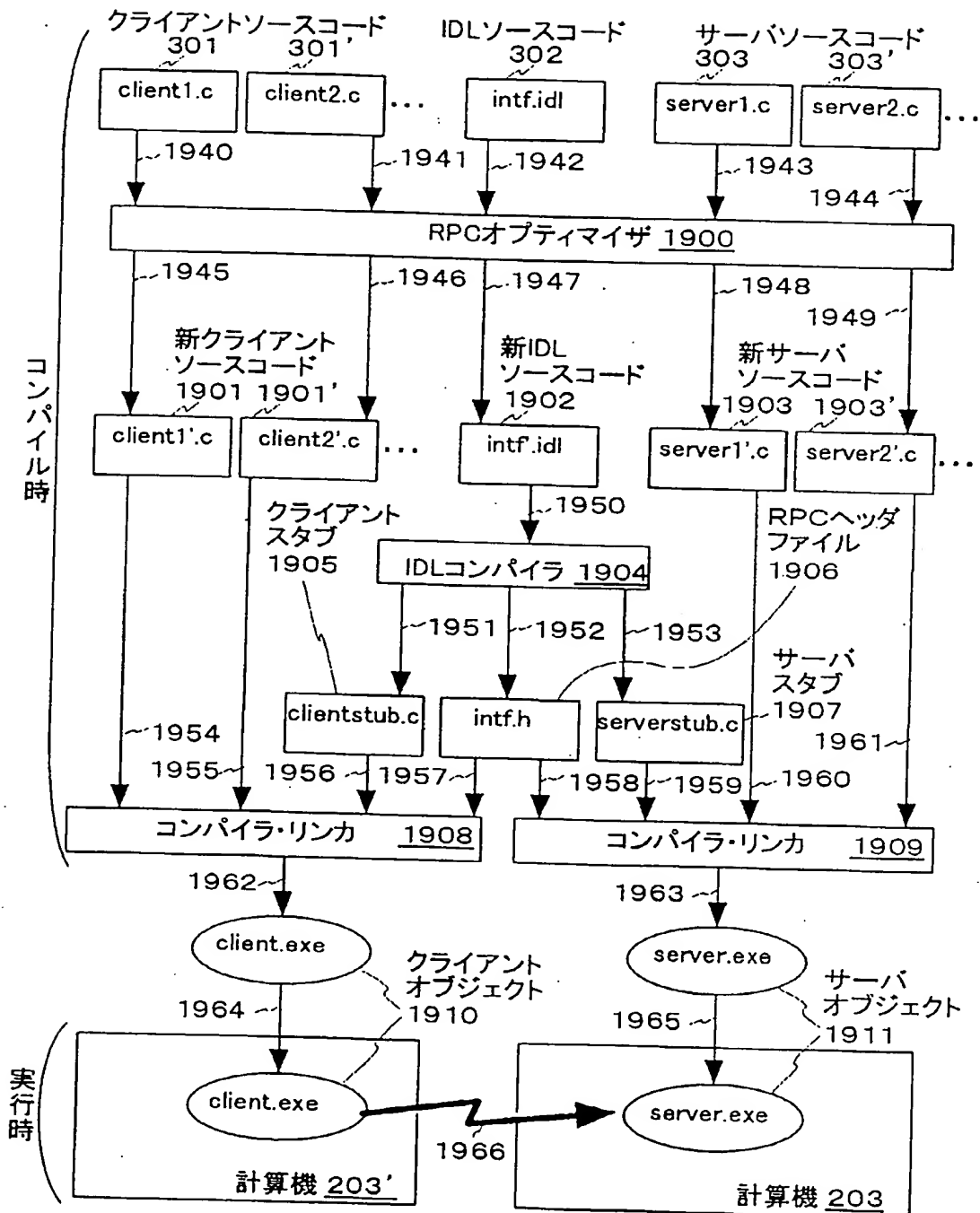
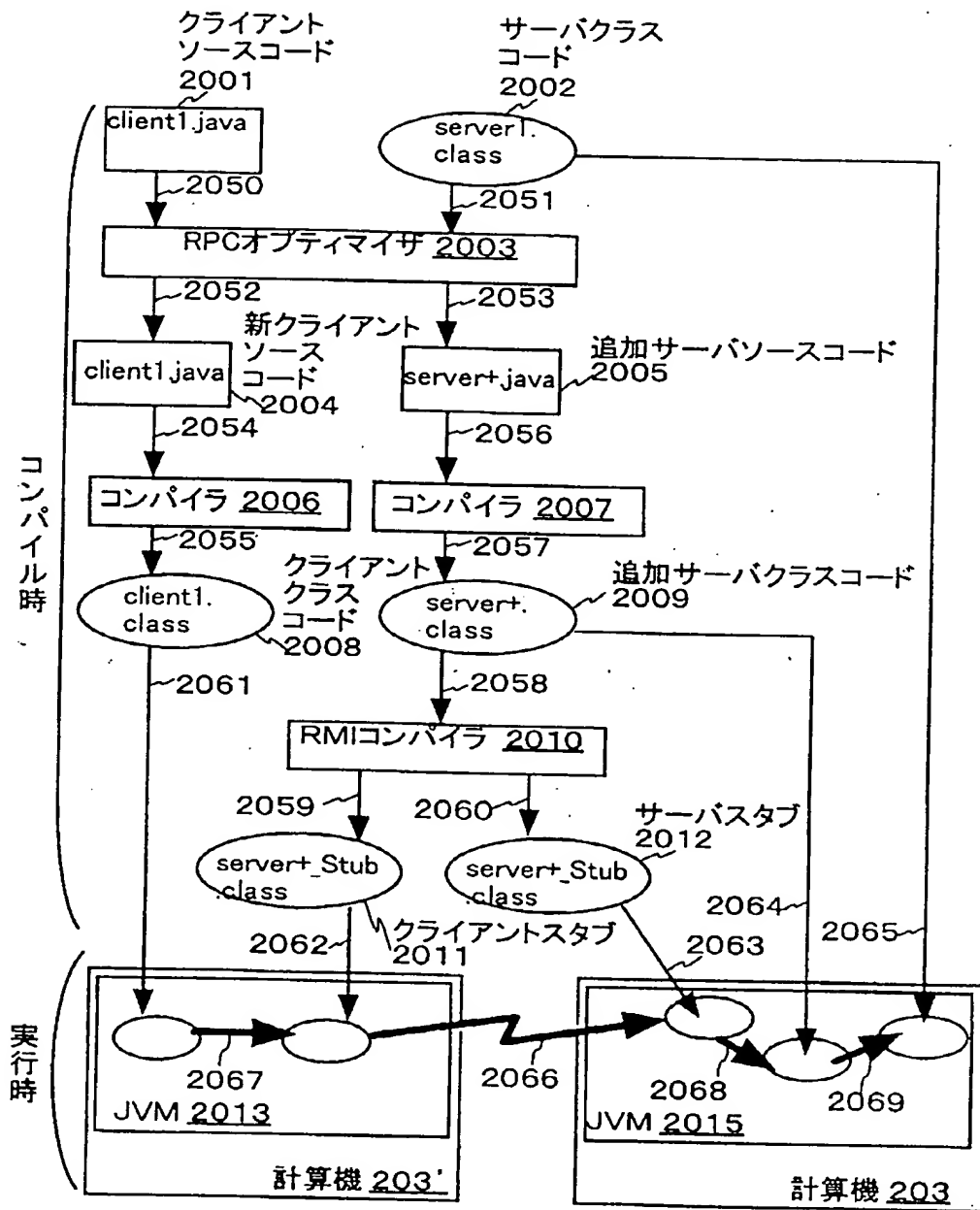




図20



【図21】

図21

